Jms Control Overview

Table of contents

1. Overview

The JMS control provides for sending messages to a queue or topic destination. It is an extensible control where sub-classes are bound to specific queues/topics and methods may be defined to send messages of specific types with specific properties and headers. The queue connections are transparently managed by the controls relieving the developer of that responsibility.

In the example below, the OrderQueue control class has one submitOrder() method that takes an Order object as the body and a string that sets the 'DeliverBy' property in the javax.jms.ObjectMessage to be sent to the queue.orders JMS queue.

```
@ControlExtension
@JMSControl.Destination(sendJndiName="queue.orders",jndiConnectionFactory="weblogic.jws
public interface OrderQueue extends JMSControl
 {
    public class Order implements java.io.Serializable
    {
        public Order()
        {

        }
        public Order(int buyer,String[] list)
        {
            buyerId = buyer;
            itemList = list;

        }
        private int buyerId;
        private String[] itemList;
    }

    public void submitOrder(Order order,@Property(name="DeliverBy") String
deliverBy);
}
```

2. Control Methods

The methods for the jms-control are:

Method	Description
getSession()	Get the queue/topic session.
getDestination()	Get the queue/topic destination.
getConnection()	Get the queue/topic connection.

setHeaders(Map)	Set the headers from the given map. The keys should be of type HeaderType or equivalent strings. Used only for the next message send. See table below for valid values.
setHeader(HeaderType,Object)	Set a header. Used only for the next message send.
setProperties(Map)	Set the properties on the message using the map of string/object pairs. Used only for the next message send.
setProperty(String,Object)	Set a property. Used only for the next message send.

The methods of the extension control-classes correspond to sending a message to a topic/queue, e.g.

```
send<message-type>(...)
```

3. Header Types

The table below defines the valid values for header types passed into setHeader() or setHeaders():

JMS Message Method	HeaderType/String	Allowed Value Types
setJMSType()	JMSType	String
setJMSCorrelationID()	JMSCorrelationID	String or byte[]
setJMSExpiration()	JMSExpiration	String valued long or Long
setJMSPriority()	Priority	String valued int or Integer

All other values are silently ignored (for backward compatibility).

4. Extension Class Annotation

The one TYPE/FIELD level annotation is:

- @Destination - defines the destination of the message, the message type and connection related attributes.

The defined annotation attributes for the @Destination element are:

Attribute	Value	Description
sendJndiName	string	JNDI name of the queue or

		topic. Required.
sendCorrelationProperty	string	The correlation property to be used for message sent. Default is empty which signifies that the JMS correlation header is to be used. Optional.
connectionFactoryJndiName	string	JNDI name of the connection factory. Required
transacted	boolean	True if en-queuing is under transactional semantics of the enclosing container. Default is true. See JMS documentation on transactional semantics of en-queueing and de-queueing.
acknowledgeMode	enum AcknowledgeMode	The acknowledgement strategy, one of Auto, Client, DupsOk. Default is Auto. See JMS API documentation on javax.jms.Session.AUTO_ACKNOWLEDGE/CLIE for more information
sendType	enum DestinationType	Values are Auto, Queue and Topic. If Auto, then the type is determined by the destination named by the sendJndiName attribute. Default is Auto.
jndiContextFactory	string	The class name of the jndi-context-factory. Default is none.
jndiProviderURL	string	The provider URL for JNDI. Default is none.

The method and parameters can be annotated with the annotation types in the next section.

5. Extension Method Annotation

The jms-control is intended to be extended. One or more methods may be defined that send messages to the given destination may be annotated with:

- @Message(message-type) or @Message or nothing.
- @Priority - the int valued attribute contains a JMS priority (0-9). If not given then the default for the provider is used.

- @Expiration - the long valued attribute contains a JMS expiration in milliseconds. If not given then the default for the provider is used.
- @Delivery - the DeliveryMode valued attribute determines the delivery mode of the message. If not given then the default for the provider is used.
- @Type - the string valued attribute determines the JMS type.
- @CorrelationId - the string valued attribute determines the correlation id.
- @Properties(PropertyValue[]) - One or more string/int/long valued properties to be added to the message. PropertyValue has the string valued attributes 'name', 'value' and class valued 'type'. The allowed values for 'type' are String.class, Integer.class and Long.class. If not given then 'String.class' is assumed.

The message-type value is a MessageType enumerated value. Values are: Auto, Object, Bytes, Text, Map and JMSMessage. If not given or no message-type string then the default is Auto. If Auto then the type of JMS message is determined by the type of the body passed in. If the body is a String or XmlObject then a TextMessage is sent, if the body is a byte[] then a StreamMessage is sent, if the body is a Map then a MapMessage is sent, if the body is a JMSMessage then it is sent, otherwise if the body is Serializable then an ObjectMessage is sent. Any other type results in a control exception.

The values of the DeliveryMode enumerated value are: NonPersistent, Persistent and Auto where Auto is the default.

6. Extension Method Parameter Annotation

Parameters to a send message method may be annotated:

- @Property (name= blah) - the parameter contains the value of the property blah.
- @Priority - the int or integer valued String parameter contains a JMS priority (0-9). If not given then the method-level annotation is used if given else the default for the provider is used.
- @Expiration - the long or integer valued String parameter contains a JMS expiration in milliseconds. If not given then the method-level annotation is used if given else the default for the provider is used.
- @Delivery - the DeliveryMode valued parameter determines the delivery mode of the message. If not given then the method-level annotation is used if given else the default for the provider is used.
- @Type - the string valued parameter determines the JMS type.
- @CorrelationId - the string valued parameter determines the correlation id.

These annotations denote which parameter is to be the body of the message and zero or more properties to be set in the message respectively.